

A Visualisation and Analysis Toolkit for Universal Graph Representation of Stabilizer Codes

Francesco Decataldo¹[0009–0006–7349–391X], Carla Piazza¹[0000–0002–2072–1628],
and Riccardo Romanello¹[0000–0002–2855–1221]

University of Udine, Udine, Italy
`decataldo.francesco@spes.uniud.it`
`{riccardo.romanello, carla.piazza}@uniud.it`

Abstract. Stabilizer codes are among the most widely used classes of quantum error-correcting codes (QECCs). Traditionally, stabilizer codes are described through their stabilizer generators. However, this approach can obscure structural properties relevant to decoding and distance analysis. The universal graph representation (UGR) provides an alternative graph-theoretic description in terms of a semi-bipartite graph. This representation translates several code-theoretic tasks, including distance computation and syndrome decoding, into graph games. Nevertheless, computing the UGR of a stabilizer code requires a non-trivial reduction pipeline involving encoder synthesis and ZX calculus rewriting. In this paper, we present an automated toolkit for computing, visualising, and analysing UGRs of stabilizer codes. The toolkit features an interactive web interface for inspecting the resulting graph structure and an exact MiniZinc model for computing the code distance. By making the UGR reduction pipeline executable and accessible, the toolkit supports experiments with stabilizer codes and their graph representations.

Keywords: Stabilizer codes · Universal Graph Representation · ZX calculus · Toolkit

1 Introduction

Quantum Error Correction (QEC) is essential for realizing scalable fault-tolerant quantum computers. The robustness and efficiency of fault-tolerant quantum computation largely depend on the properties of the adopted quantum error-correcting code (QECC), such as rate, distance, and decoding complexity. Among QECCs, stabilizer codes form one of the most widely studied and practically used classes. Traditionally, stabilizer codes are specified and analysed through algebraic representations. Recently, the universal graph representation (UGR) was introduced as a graph-theoretic perspective on stabilizer codes [9]. Based on the ZX calculus [13] and the graph states formalism [6], this representation maps algebraic properties of the code into structural properties of a semi-bipartite graph. This graph structure encodes algorithmically relevant properties of the

code, transforming tasks such as distance computation and decoding strategies into a classical game on a graph. However, computing the UGR from stabilizer generators is a complex process involving circuit synthesis and ZX calculus rewriting rules. Currently, there is a lack of accessible software bridging the gap between this theoretical framework and practical analysis by QEC researchers. This makes it difficult to experiment with UGRs or compare graph structures across code families.

In this paper, we present an automated software toolkit that computes the universal graph representation (UGR) of any stabilizer code. In particular, it allows users to interactively explore the resulting graph and gain insight into its structural properties. Our main contributions are:

1. **Automated reduction:** We implement and automate the reduction from the generators of a stabilizer code to the UGR, with `Stim` and `PyZX` libraries.
2. **Web interface:** We provide a web interface to allow users to generate and interact with UGRs of predefined, custom, and randomly generated codes.
3. **Distance model:** We integrate a MiniZinc constraint-programming model for computing the code distance directly from the UGR.

The paper is organized as follows: Section 2 introduces the mathematical notation and background on stabilizer codes and ZX calculus. Section 3 formally defines the universal graph representation and explains how the code distance can be calculated from the graph’s structure. Section 4 describes the algorithmic reduction from a stabilizer code to its UGR following the construction of Khesin et al. [9]. Section 5 describes additional toolkit features and reports experimental results for the distance model. Section 6 presents examples on well-known quantum codes, including topological codes. The toolkit is available online¹.

2 Preliminaries

We first introduce the basic notions used throughout the paper. As a shorthand, given a single-qubit operator U , we write U_i for the n -qubit operator that applies U to qubit i and acts as the identity on all other qubits. Throughout the paper, qubits are numbered from 1 to n .

$$U_i = I^{\otimes(i-1)} \otimes U \otimes I^{\otimes(n-i)}.$$

Similarly, if U is a two-qubit gate, then $U_{i,j}$ denotes the n -qubit operator that applies U to qubits i and j and acts as the identity elsewhere.

2.1 Choi–Jamiołkowski isomorphism

The main reduction is based on the possibility of representing a linear map (and thus a unitary transformation) as a quantum state and vice-versa. This can be done as a consequence of the Choi–Jamiołkowski isomorphism [11].

¹ https://github.com/fradeca01/pyzx_ugrep

Theorem 1 (Choi–Jamiołkowski isomorphism). *Let $\mathcal{H}_A$ and $\mathcal{H}_B$ be finite-dimensional Hilbert spaces, and fix an orthonormal basis $\{|i\rangle\}_{i=0}^{\dim(\mathcal{H}_A)-1}$ of $\mathcal{H}_A$. Define the unnormalized state*

$$|\Omega_A\rangle \equiv \sum_{i=0}^{\dim(\mathcal{H}_A)-1} |i\rangle \otimes |i\rangle$$

Then the map

$$L \mapsto (I_A \otimes L) |\Omega_A\rangle$$

is a linear bijection between $\text{Lin}(\mathcal{H}_A, \mathcal{H}_B)$ and $\mathcal{H}_A \otimes \mathcal{H}_B$, where $\text{Lin}(\mathcal{H}_A, \mathcal{H}_B)$ denotes the set of linear maps from $\mathcal{H}_A$ to $\mathcal{H}_B$.

2.2 Stabilizer Codes

We now review the fundamental concepts of the stabilizer formalism and quantum error-correcting codes [4]. A Quantum Error Correcting Code (QECC) is a method of protecting quantum information from noise by embedding it into a larger Hilbert space.

Definition 1 (QECC). *A QECC is a 2^k -dimensional subspace $\mathcal{C}$ (the code space) of the n -qubit Hilbert space $\mathcal{H}^n$, where $n \geq k$.*

We now consider the subclass of QECCs called stabilizer codes. Let $\mathcal{P}_1$ be the single-qubit Pauli group, formed by Pauli matrices I, X, Y, Z along with the phase factors $\{\pm 1, \pm i\}$.

We generalize the notion of *Pauli group* to the n -qubit case.

Definition 2 (Pauli Group). *The n -qubit Pauli group $\mathcal{P}_n$ is the n -fold tensor product of the single-qubit Pauli group:*

$$\mathcal{P}_n \equiv \{Q_1 \otimes \cdots \otimes Q_n \mid Q_i \in \mathcal{P}_1\}. \quad (1)$$

Let $P = Q_1 \otimes \cdots \otimes Q_n \in \mathcal{P}_n$ where $Q_i \in \mathcal{P}_1$. The weight of P is the number of non-identity tensor factors:

$$\text{wt}(P) = |\{i \in \{1, \dots, n\} : Q_i \neq I\}|.$$

Stabilizer codes are a class of quantum codes that can be defined through a commuting set of Pauli operators. In particular, consider the following class of subgroups of $\mathcal{P}_n$, the *stabilizer groups*.

Definition 3 (Stabilizer group). *A stabilizer group $\mathcal{S}$ is a subgroup of $\mathcal{P}_n$ that is abelian and does not contain $-I$.*

In particular it is possible to prove the following.

Lemma 1. *Given a stabilizer group $\mathcal{S}$, every element of $\mathcal{S}$ has eigenvalues ± 1 .*

As a consequence of the lemma above we can define stabilizer codes [4].

Definition 4 (Stabilizer Code). An $[[n, k, d]]$ stabilizer code is a subspace $\mathcal{C} \subseteq \mathcal{H}^n$ of dimension 2^k , defined as the common $+1$ eigenspace of all the elements of a stabilizer subgroup $S(\mathcal{C}) \subset \mathcal{P}_n$ of rank $n - k$.

$$\mathcal{C} = \{|\psi\rangle \mid P|\psi\rangle = |\psi\rangle \quad \forall P \in S(\mathcal{C})\}.$$

The distance d is defined as the minimum weight of a Pauli operator that commutes with every element of $S(\mathcal{C})$ but is not itself proportional to an element of $S(\mathcal{C})$; equivalently, it is the minimum weight of a Pauli operator P that acts non-trivially on the encoded states but preserves the codespace.

By Definition 4, if the stabilizer group has rank n , then $\mathcal{C}$ has dimension 1 and therefore defines a unique quantum state.

Definition 5 (Stabilizer State). A stabilizer state is a quantum state that is uniquely defined by a stabilizer group $S \subset \mathcal{P}_n$ of rank n .

We now consider the set of unitary transformations that map Pauli operators to Pauli operators, thereby preserving stabilizer groups [5].

Definition 6 (Clifford Group). The n -qubit Clifford group $\mathcal{C}_n$ is defined as:

$$\mathcal{C}_n \equiv \{U \in \text{U}(2^n) \mid UPU^\dagger \in \mathcal{P}_n \quad \forall P \in \mathcal{P}_n\}.$$

This group is generated by the Hadamard, Phase and CNOT gates. We will also use local Clifford operators, namely Clifford operators restricted to single qubits [6].

Definition 7 (Local Clifford operator). A local Clifford operator is a tensor product of single-qubit Clifford operators. Each single-qubit Clifford operator is generated by the gates H and S .

An encoding isometry for an $[[n, k]]$ stabilizer code $\mathcal{C}$ is an embedding $V : \mathcal{H}^k \rightarrow \mathcal{C} \subset \mathcal{H}^n$ such that $V^\dagger V = I$. In particular, the following lemma holds.

Lemma 2. Two isometries V_1 and V_2 encode the same stabilizer code if and only if $V_2 = V_1 U$ for some unitary U acting on the input space.

An isometry V can also be implemented as a two-step transformation. We first add $n - k$ ancilla qubits and then apply the actual encoding through a unitary transformation U_{enc} .

$$V|\psi\rangle = U_{\text{enc}} \left(|0\rangle^{\otimes(n-k)} \otimes |\psi\rangle \right) \quad \forall |\psi\rangle \in \mathcal{H}_k. \quad (2)$$

In the following, we refer to U_{enc} as an *encoder*. Consider a set of generators $\{S_1, \dots, S_{n-k}\}$ for the stabilizer group of $\mathcal{C}$. Since $|0\rangle$ is stabilized by Z , any encoder maps the subspace stabilized by $\{Z_1, \dots, Z_{n-k}\}$ to the code space $\mathcal{C}$.

Lemma 3 (Encoder). *An encoder for a $[[n, k, d]]$ stabilizer code $\mathcal{C}$ generated by $\{S_1, \dots, S_{n-k}\}$ can be described by any Clifford transformation U_{enc} satisfying:*

$$U_{\text{enc}} Z_i U_{\text{enc}}^\dagger = S_i, \quad i = 1, \dots, n - k.$$

where Z_i is $\underbrace{I \otimes \dots \otimes Z}_i \otimes \dots \otimes I$.

In other words, every completion of the Clifford transformation U_{enc} satisfying the required Pauli commutation relations gives a valid encoder for the code $\mathcal{C}$.

2.3 ZX Calculus

The ZX calculus is a graphical language for representing linear maps, equipped with a set of rewriting rules for manipulating diagrams [1, 13]. In this paper, we focus on diagrams that represent Clifford operators and we state the definitions only for the Clifford fragment. All ZX diagram equalities are considered up to a non-zero scalar since normalization factors are irrelevant for our reduction. The building blocks of ZX diagrams are *spiders*. They can be either Z or X spiders.

Definition 8 (Clifford $Z(X)$ -spider). *A Clifford Z -spider (X -spider) with n inputs, m outputs and phase $\alpha \in \{0, \frac{\pi}{2}, \pi, \frac{3}{2}\pi\}$ is defined as a linear map:*

$$\begin{aligned} Z : \begin{array}{c} \text{---} \text{---} \text{---} \text{---} \text{---} \text{---} \\ \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \end{array} \begin{array}{c} \text{---} \text{---} \text{---} \text{---} \text{---} \text{---} \\ \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \end{array} \alpha &\equiv \underbrace{|0 \dots 0\rangle}_m \underbrace{\langle 0 \dots 0|}_n + e^{i\alpha} \underbrace{|1 \dots 1\rangle}_m \underbrace{\langle 1 \dots 1|}_n \\ X : \begin{array}{c} \text{---} \text{---} \text{---} \text{---} \text{---} \text{---} \\ \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \end{array} \begin{array}{c} \text{---} \text{---} \text{---} \text{---} \text{---} \text{---} \\ \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \end{array} \alpha &\equiv \underbrace{|+\dots+\rangle}_m \underbrace{\langle +\dots+|}_n + e^{i\alpha} \underbrace{|-\dots-\rangle}_m \underbrace{\langle -\dots-|}_n \end{aligned}$$

Clifford ZX diagrams are defined by connecting wires of different spiders (horizontal composition) or by stacking one spider on top of another (vertical composition). It can be proved that spider wires can be *bent* and, as long as the ordering of inputs and outputs does not change, nodes can be moved across the space freely without changing the represented linear map. The Hadamard gate can be expressed in terms of X and Z spiders. We use the following notation.

Definition 9 (ZX Hadamard). *The Hadamard gate in the ZX calculus is represented by a dashed or boxed edge.*

$$\begin{aligned} \text{---} \square \text{---} &= \text{-----} \\ &\equiv \text{---} \bigcirc \frac{\pi}{2} \bigcirc \frac{\pi}{2} \bigcirc \frac{\pi}{2} \text{---} \end{aligned}$$

A wire decorated with a Hadamard gate is referred to as a Hadamard wire (or Hadamard edge).

The ZX calculus consists of a finite set of basic rules for manipulating ZX diagrams; we refer the reader to [13] for an introductory presentation on ZX calculus rules. Clifford ZX diagrams with no inputs represent stabilizer states, up to a non-zero scalar [13]. Within this class, we now distinguish a particular graph-like form, which corresponds to the standard notion of graph states with local Clifford operations [6].

Definition 10 (Extended graph state form). *A Clifford ZX diagram is in extended graph state form if and only if it satisfies the following conditions:*

- **State:** *The diagram has no inputs; hence every free edge is an output edge;*
- **Z spiders:** *All spiders in the diagram are Z-spiders;*
- **Free edges:** *Every spider is incident to exactly one free edge, i.e. an edge connected to only one spider;*
- **Hadamard edges:** *Every edge between spiders is a Hadamard edge;*
- **Graph form:** *There are no self-loops or parallel edges.*

In particular the following holds:

Theorem 2 ([2, 12]). *Every Clifford ZX diagram with no inputs (i.e. every diagram representing a stabilizer state) can be transformed into extended graph state form using the rules of the ZX calculus.*

However, such a representation is not unique, i.e., there may be several Clifford ZX diagrams in extended graph state form that represent the same stabilizer state. To remove this ambiguity, a canonical form was introduced in [7].

Definition 11 (ZX graph state canonical form). *We say that a Clifford ZX diagram in extended graph state form is in canonical form if and only if:*

- **Hadamard ordering:** *Whenever a vertex has a Hadamard free edge, it is not connected to any lower-numbered node, where the node order is induced by the order of the output edges.*
- **Local Cliffords:** *if a node has a Hadamard free edge, its phase is not $\frac{3}{2}\pi$.*

We now recall two rules to manipulate Clifford ZX diagrams in extended graph state form, which are at the core of the reduction (Section 4). These rules are based on the *local complementation* operation.

Definition 12 (Local complementation). *Given a graph $G = (V, E)$, local complementation at a vertex $x \in V$ is the graph transformation that produces a new graph $L_x(G) = (V, E')$ such that the subgraph induced by the neighbourhood of x , defined as $N(x)$, is complemented (where $x \notin N(x)$), i.e., $\forall y, z \in N(x)$:*

$$(y, z) \in E' \quad \Leftrightarrow \quad (y, z) \notin E$$

and all the other edges remain unchanged.

An example of local complementation on a graph is shown in Figure 2. Local complementation also gives rise to pivoting along an edge.

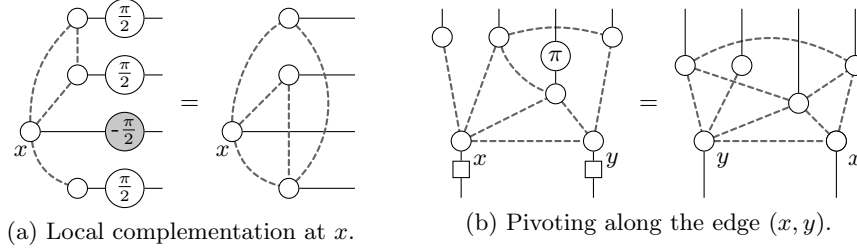


Fig. 1. Examples of the local complementation and pivot rules for Clifford ZX-diagrams in extended graph state form.

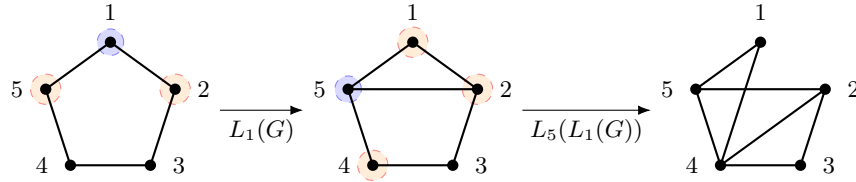


Fig. 2. A sequence of local complementations on a graph G : first at vertex 1, producing $L_1(G)$, and then at vertex 5, producing $L_5(L_1(G))$.

Definition 13 (Pivoting). Given a graph $G = (V, E)$ and an edge $(x, y) \in E$, pivoting along (x, y) is the transformation:

$$P_{(x,y)}(G) = L_x(L_y(L_x(G)))$$

As proved in [3], it is possible to local complement around a vertex or pivot along an edge of a Clifford ZX diagram by applying rules of the ZX calculus. In particular, the following result holds.

Theorem 3. Let D be a Clifford ZX diagram in extended graph state form, and let G be its underlying graph. The following equivalences hold:

- For any vertex/spider x , the local complementation equivalence relates D to the diagram whose underlying graph is $L_x(G)$. The equivalence is obtained by adding a phase $\frac{\pi}{2}$ to each neighbour of x and attaching an X-spider with phase $-\frac{\pi}{2}$ to x , as shown in Figure 1a.
- For any edge (x, y) , the pivot equivalence relates D to the diagram whose underlying graph is $P_{(x,y)}(G)$. The equivalence swaps the roles of x and y , toggles the types of their free edges, and adds a phase π to each common neighbour of x and y , as shown in Figure 1b.

We will use these rules extensively in the algorithm that computes the universal graph representation.

3 Universal Graph Representation

This section reviews the theoretical framework of the universal graph representation introduced by [9]. This formalism provides a graph-theoretic perspective on quantum stabilizer error-correcting codes, expressing code properties, such as code distance, in graph-theoretic terms. Given an $[[n, k, d]]$ stabilizer code $\mathcal{C}$, the goal is to represent it canonically as a semi-bipartite graph $(G, \mathcal{I}, \mathcal{O})$. The graph has k input vertices and n output vertices, corresponding respectively to logical and physical qubits. We label the vertices from 1 to $n + k$ with $\mathcal{I} = \{1, \dots, k\}$ and $\mathcal{O} = \{k + 1, \dots, n + k\}$. Unlike in a standard bipartite graph where all edges must be between the two partitions, a semi-bipartite graph relaxes this requirement for one set: while the input set $\mathcal{I}$ is required to be an independent set (no internal edges), the output set $\mathcal{O}$ may contain internal edges. The graph is obtained from the underlying graph of a Clifford ZX diagram in ZX canonical form (ZXCF) [9], which represents the stabilizer code $\mathcal{C}$. ZXCF extends the canonical form for diagrams with no inputs (Definition 11) to Clifford ZX diagrams with both inputs and outputs.

Definition 14 (ZX canonical form). *Consider a Clifford ZX diagram whose input and output edges are labelled in increasing order, with input edges assigned lower labels than output edges. The diagram is said to be in ZX canonical form (ZXCF) if and only if it satisfies all the conditions of Definition 10 and Definition 11, except for the **State** condition of Definition 10, and also satisfies the following additional conditions:*

1. **Semi-bipartite structure:** *The vertex set can be partitioned into two disjoint subsets: input nodes $\mathcal{I}$ (incident to input edges) and output nodes $\mathcal{O}$ (incident to output edges), such that no edge exists between two nodes in $\mathcal{I}$.*
2. **Echelon form:** *The binary adjacency matrix between nodes in $\mathcal{I}$ and $\mathcal{O}$ is in row-reduced echelon form (RREF) over $\mathbb{F}_2$. We denote by $\mathcal{P} \subseteq \mathcal{O}$ the set of output nodes corresponding to the pivot columns of this adjacency matrix.*
3. **Pivot edges:** *No two nodes in $\mathcal{P}$ are connected by an edge;*
4. **Pivot local Cliffords:** *Nodes in $\mathcal{P}$ must not have a local Clifford operator on their free edges. Equivalently, each such node has phase 0 and its free edge is not a Hadamard edge.*

Khesin et al. [9] show that there is a bijection between ZXCF diagrams and stabilizer codes. We can now define the universal graph representation.

Definition 15. *The universal graph representation (UGR) of a stabilizer code $\mathcal{C}$ is the triple $(G, \mathcal{I}, \mathcal{O})$ obtained from the corresponding ZXCF diagram, where G is its underlying graph and $\mathcal{I}, \mathcal{O}$ are the sets of input and output nodes, respectively. The local Clifford operations of the ZXCF diagram are not included in the UGR.*

The reduction algorithm described in the next section constructs the corresponding ZXCF diagram and extracts its UGR. It is important to note that the local Clifford operations are discarded when passing from the ZXCF to the UGR.

Therefore, the UGR represents the code up to local Clifford equivalence. This omission does not affect the parameters n , k , or the distance d , since local Clifford operations map Pauli errors to Pauli errors affecting the same qubits.

The UGR provides a graph-theoretic framework for studying stabilizer codes, as it expresses several code properties in terms of graph-theoretic quantities. For instance, bounds on the code distance can be derived from vertex degrees.

Khesin et al. [9] also introduced a family of games associated with the UGR. These games capture several code properties and algorithmic tasks related to the code. Specifically, the following problems can all be formulated as instances of Quantum Lights Out (QLO) games:

- Computation of the code distance.
- Search for a minimum-weight generating set of the stabilizer group.
- Decoding procedures, where a QLO strategy specifies which moves to apply to the light configuration associated with a given syndrome.

In this work, we focus on the distance-QLO instance, which corresponds to code-distance computation.

Definition 16 (Distance-QLO game). *Consider the following single-player game on a semi-bipartite graph $(G = (V, E), \mathcal{I}, \mathcal{O})$ (the UGR of a code $\mathcal{C}$). Each vertex $v \in V$ has a light that can be either **on** or **off**. The game is played in two sequential rounds.*

- Start with all lights **off**.
- During the first round, it is possible to flip any light in $\mathcal{I}$.
- During the second round, each move consists of one of the following actions:
 - flip the light at a vertex $v \in \mathcal{O}$;
 - select a vertex $v \in \mathcal{O}$, flip all neighbouring lights, and then permanently turn v **off**, so that it cannot be turned on again by later moves.

*The goal is to turn **off** all lights in $\mathcal{O}$ using the minimum possible number of second-round moves.*

It has been proved that the minimal number of moves made in the second round to turn **off** all the lights in $\mathcal{O}$ in distance-QLO is the code distance [9]. Since computing the distance of a stabilizer code is NP-hard [8], solving the corresponding optimal distance-QLO problem is NP-hard as well. This characterization motivates the MiniZinc model described in Section 5, which encodes distance-QLO directly as a constraint-programming problem.

4 Reduction

In this section we describe the reduction algorithm, the main contribution of our toolkit. Given a set of generators for a stabilizer code $\mathcal{C}$, it outputs the corresponding universal graph representation. Our implementation follows the reduction introduced in [9], with some modifications to the initial stage that are described in Steps A–D below. The reduction consists of the following steps:

- A. Compute a Clifford operation representing an encoder for the code $\mathcal{C}$.
- B. Compute the stabilizer state corresponding to the previous Clifford operation using the Choi–Jamiołkowski isomorphism (Theorem 1).
- C. Synthesize a circuit that prepares the stabilizer state in a form that corresponds to the extended graph state form (Definition 10, Theorem 2)
- D. Convert the circuit into a ZX diagram in extended graph state form (Definition 10).
- E. Reduce the ZX diagram in extended graph state form to its (unique) canonical form (Definition 11, cf. [7]);
- F. Convert the state back to an operator via the Choi–Jamiołkowski isomorphism (Theorem 1);
- G. Transform the ZX diagram into ZXCF by enforcing the conditions of Definition 14. Extract the graph structure from the ZXCF to obtain the UGR (Definition 15).

The complete pipeline is exposed through the `stabilizers_to_UGR()` function of our library. Steps A–D transform the input stabilizer code into a ZX diagram in extended graph state form Definition 10. These steps are implemented in the `ugr.stabilizers_to_ZX_graph(...)` method of our library. Unlike the procedure in [9], which first synthesizes an encoding circuit, converts it into a ZX diagram, and subsequently reduces the diagram to extended graph state form, our implementation operates directly on the Clifford tableau and synthesizes a ZX diagram which is in extended graph state form.

Steps E–F canonicalize this diagram and reinterpret it as an encoder. These steps are implemented in the `ugr.graph_to_ZXCF(...)` method. Step G canonicalizes the diagram also with respect to equivalences on the inputs to obtain the ZXCF and hence the UGR. This is implemented in the `ugr.ZXCF_to_UGR(...)` method of our library.

4.1 Step A: Clifford Encoder

The first step constructs a Clifford encoder U_{enc} from the stabilizer generators (Lemma 3). These generators specify the images of $\{Z_1, \dots, Z_{n-k}\}$, whereas a full Clifford tableau requires the images of the generating set $\{X_i, Z_i\}_{i=1}^n$. Any completion that is independent and respects the commutation relations of the generators defines a valid Clifford encoder and may be chosen arbitrarily. We use the `stim.Tableau.from_stabilizers(...)` method from the `Stim` library to compute such a completion. This method has complexity $O(n^3)$.

4.2 Step B: Construct the Corresponding Stabilizer State

The second step of the algorithm applies the Choi–Jamiołkowski isomorphism (Theorem 1) to obtain a stabilizer state from the encoder U_{enc} obtained in the previous step. The Clifford operation U_{enc} can be described by its action on a generating set for the Pauli group:

$$\begin{cases} U_{\text{enc}} Z_i U_{\text{enc}}^\dagger = S_i^Z, \\ U_{\text{enc}} X_i U_{\text{enc}}^\dagger = S_i^X, \end{cases} \quad i = 1, \dots, n.$$

Since U_{enc} encodes a $[[n, k, d]]$ code, we know that its first $n - k$ inputs are always $|0\rangle^{\otimes(n-k)}$. Consider the linear map V of Equation (2):

$$V |\psi\rangle = U_{\text{enc}}(|0\rangle^{\otimes(n-k)} \otimes |\psi\rangle) \quad \forall |\psi\rangle \in \mathcal{H}^k$$

By Theorem 1, we can associate to V the state $(I_k \otimes V) |\Omega_k\rangle$, where:

$$|\Omega_k\rangle = \sum_{i=0}^{2^k-1} |i\rangle \otimes |i\rangle$$

is the unnormalized maximally entangled state. Therefore, the associated state can be described as:

$$\begin{aligned} |E\rangle &= (I_k \otimes V)(|\Omega_k\rangle) = (I_k \otimes V) \sum_{i=0}^{2^k-1} |i\rangle \otimes |i\rangle \\ &= (I_k \otimes U_{\text{enc}}) \left(\sum_{i=0}^{2^k-1} |i\rangle \otimes |0\rangle^{\otimes(n-k)} \otimes |i\rangle \right) \end{aligned}$$

Our goal is to compute a set of stabilizer generators for $|E\rangle$. To do this, we first compute stabilizer generators for:

$$|S\rangle = \sum_{i=0}^{2^k-1} |i\rangle \otimes |0\rangle^{\otimes(n-k)} \otimes |i\rangle$$

and then evolve the group through $(I_k \otimes U_{\text{enc}})$ by conjugation. Since $|S\rangle$ is a state of $n + k$ qubits, we need $n + k$ stabilizers. We get the first $n - k$ by using Pauli Z for the $|0\rangle^{\otimes(n-k)}$ part of the state:

$$\{Z_{k+1}, \dots, Z_n\}$$

The remaining $2k$ stabilizers are given by the k Bell pairs formed by the first k qubits and the last k qubits:

$$\{Z_i Z_{n+i}, X_i X_{n+i}\}_{i=1}^k$$

These $n + k$ independent, commuting Pauli operators generate the stabilizer group of $|S\rangle$. We then evolve this set under conjugation by $I_k \otimes U_{\text{enc}}$ to obtain a stabilizer generating set for $|E\rangle$. This evolution can be done in time $O((n + k)^2)$.

4.3 Step C: Synthesize a Circuit Preparing The State

By Theorem 2, any stabilizer state can be represented, up to a non-zero scalar, as a Clifford ZX diagram in extended graph state form. In circuit terms, this means that the state can be prepared from $|+\rangle^{\otimes(n+k)}$ by applying a layer of CZ gates, which determines the underlying graph, followed by a layer of single-qubit Clifford gates [13]. In our implementation this representation is obtained using `stim.Tableau.to_circuit(method="graph_state")`. The procedure runs in $O((n + k)^3)$ and outputs a circuit with $O((n + k)^2)$ gates in the worst case.

4.4 Step D: Convert to a ZX diagram In Extended Graph State Form

This step of the algorithm takes the circuit obtained in the previous step and transforms it into a ZX diagram D . This step is straightforward since every Clifford gate can be translated into a constant-size ZX diagram. The resulting ZX diagram is in extended graph state form (Definition 10). The conversion can be done in time $O((n+k)^2)$. This follows because the circuit has $O((n+k)^2)$ gates in the worst case, and each gate is converted into a constant number of spiders and edges. In our implementation, this is done using the PyZX library.

4.5 Step E: Reduction to Canonical Form

Let D be the Clifford ZX diagram in extended graph state form produced in the previous step. This step reduces it to ZX graph state canonical form (Definition 11), following the algorithm of Hu–Khesin [7]. The procedure tracks the local Clifford applied at each vertex and iteratively applies local complementation and pivoting equivalences until the canonical form conditions of Definition 11 are satisfied. This step has complexity $O((n+k)^3)$ and is implemented in the `GraphState.to_canonical_form()` method of our library.

4.6 Step F: State to Operator

Given the resulting ZX diagram D in ZX graph state canonical form, we map it back to a linear map by applying the Choi–Jamiołkowski isomorphism (Theorem 1). The conversion can be done in linear time $O(n+k)$. At this point, D has k inputs and n outputs. We denote the input spiders $\mathcal{I} = \{1, \dots, k\}$ and the output spiders $\mathcal{O} = \{k+1, \dots, k+n\}$.

4.7 Step G: Reduce to the Universal Graph Representation

The last step applies unitary operations to the input qubits to transform the ZX diagram D into a ZXCF diagram. By Lemma 2, this does not change the represented code. It remains to enforce the conditions of Definition 14. The conditions of Definition 10 and Definition 11 except for the **State** condition, are already satisfied at this point of the reduction because D is in ZX graph state canonical form. We apply unitaries to the inputs to enforce the remaining conditions. In particular:

- To satisfy the **Semi-bipartite structure** condition we need to remove edges between input nodes to obtain a semi-bipartite graph. The edges can be removed without additional transformations since edges between inputs correspond to CZ applied to the inputs. Since $CZ^2 = I$, applying the same CZ to the corresponding input pair removes the edge.
- To satisfy the **Echelon form** condition we turn the adjacency matrix between input and output nodes into RREF form. This can be done using the following lemma [9].

Lemma 4. *Applying a CNOT with control i and target j , for $i, j \in \mathcal{I}$, implements the elementary row operation that adds row j to row i over $\mathbb{F}_2$ in the adjacency matrix between nodes in $\mathcal{I}$ and nodes in $\mathcal{O}$.*

Therefore, Gaussian elimination can be implemented using CNOT gates on input vertices to bring the matrix into RREF.

- To satisfy the **Pivot local Cliffords** condition we remove phases on pivot spiders. This can be achieved by applying a sequence of ZX calculus rewrites based on the local complementation rule (Theorem 3) iteratively to pivot spiders with non-zero phase.
- To satisfy the **Pivot edges** condition we remove edges between pivot vertices. This can be achieved by applying a sequence of ZX rewrites based on the pivot rule (Theorem 3) iteratively to edges between pivot vertices.

This step has complexity $O(k^2n)$ due to the Gaussian elimination algorithm. The whole algorithm has a time complexity of $O(n^3)$ since $k \leq n$.

5 Additional Toolkit Features

In this section we describe some additional features of our toolkit, beyond the core reduction algorithm presented in Section 4.

5.1 Distance Model

The toolkit includes a MiniZinc [10] model for computing the distance of a stabilizer code $\mathcal{C}$ from its UGR. The model is implemented in the file `src/qlo.mzn`. This file encodes the game of Definition 16 as a constraint-programming optimization problem. When solved to optimality, the model returns the exact distance of the stabilizer code represented by the UGR. Figure 3 reports the runtime of the MiniZinc model on randomly generated code instances; random stabilizers are generated using the `stim.Tableau.random()` method. Three independently sampled instances are used for each value of n . The experiments were run on an Azure Standard.F8s_v2 virtual machine running Ubuntu. This instance has 8 vCPUs and 16 GB of memory. Figure 3 is intended as a feasibility measurement rather than a comparison with existing distance algorithms. With $k = 5$, the largest sampled instances solved by Gecode within the one-hour timeout had $n = 38$ physical qubits.

5.2 Web Interface

We also provide a web interface for generating and interacting with UGRs. It allows users to compute and visualise UGRs via three distinct input methods:

- selecting from a set of predefined codes (Steane, Shor, and topological codes);
- providing custom stabilizer generators;
- generating random codes for exploration.

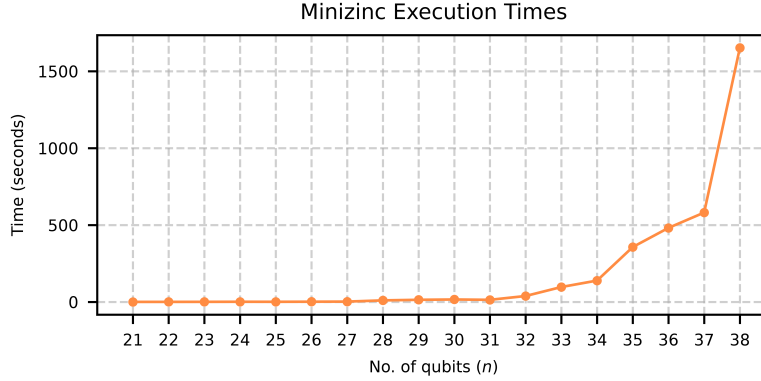


Fig. 3. MiniZinc model execution time on random codes, varying n with fixed $k = 5$. Each point is the mean over three independently generated random instances.

Users can interact directly with the graph structure by dragging vertices and adjusting the camera perspective to inspect edge connectivity. The tool enables users to inspect structural features such as the vertex degrees and edge connections. In addition, the interface leverages our constraint programming model to compute and display the exact distance when the constraint model terminates.

6 Examples

In this section, we generate and visualise the UGRs of well-known stabilizer codes. The results are shown in Figure 4. We consider the following examples:

- the Steane code, a CSS code constructed from the classical Hamming code and the smallest CSS code with distance 3 (Figure 4a);
- the five-qubit code, the smallest stabilizer code with distance 3 (Figure 4b);
- the $[[8, 2, 2]]$ and the $[[18, 2, 3]]$ toric code, respectively the smallest toric code and the smallest toric code with distance 3 (Figure 4c, Figure 4d);
- the $[[9, 1, 3]]$ rotated surface code (Figure 4e).

Notice that, while the Steane code and the Five qubit code have relatively sparse and structured UGRs, for larger topological-code instances, the resulting graphs become denser and visually more complex. These examples illustrate that UGRs can become dense even for modest topological-code instances, motivating automated generation and analysis. By automating this reduction and providing an interactive 3D environment, the toolkit allows users to inspect the degree distributions and visually distinguish the input ($\mathcal{I}$) from the output ($\mathcal{O}$) nodes. These features are relevant to the QLO game of Definition 16, where the input/output partition and vertex neighbourhoods determine the allowed moves.

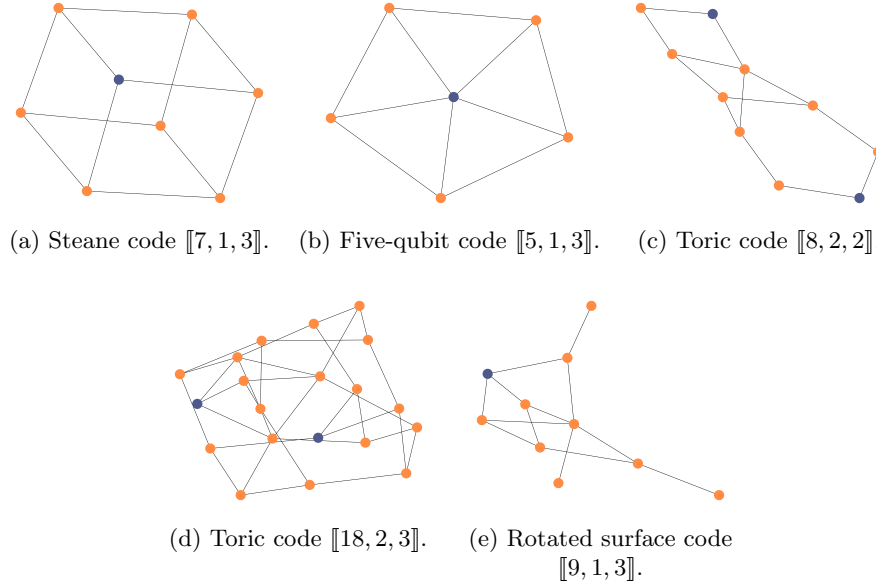


Fig. 4. Universal graph representations of selected stabilizer codes. Blue nodes are input nodes while orange nodes are output nodes.

7 Conclusions

We presented a toolkit for computing, visualising and analysing the universal graph representation of stabilizer codes proposed by Khesin et al. [9]. The implementation automates the reduction from stabilizer tableaus to UGRs and exposes an interactive visualisation interface. Furthermore, we integrated a MiniZinc constraint programming model to compute the exact distance of the code directly from the graph’s structure, together with the generation of a quantum encoding circuit. The current benchmark characterizes only the MiniZinc solver on a limited random-code sample; it does not compare distance-computation baselines or empirically benchmark the complete reduction and visualisation pipeline. Future work will address these comparisons and implement graph-based decoding algorithms derived from the QLO game. This would extend the toolkit from distance computation to QLO-based decoding experiments.

Disclosure of Interests. This work has been partially supported by INdAM-GNCS project *Algebra lineare quantistica, state preparation e compilazione di circuiti quantistici* (CUP E53C25002010001) and by the regional project QUASAR-FVG *Calcolo e simulazione quantistica: sviluppo, applicaizoni e ricerca in Friuli Venezia Giulia* (CUP G23C25001510002).

References

1. Coecke, B., Duncan, R.: Interacting quantum observables: Categorical algebra and diagrammatics. *New Journal of Physics* **13**(4), 043016 (2011). <https://doi.org/10.1088/1367-2630/13/4/043016>
2. Duncan, R., Kissinger, A., Perdrix, S., van de Wetering, J.: Graph-theoretic Simplification of Quantum Circuits with the ZX-calculus. *Quantum* **4**, 279 (2020). <https://doi.org/10.22331/q-2020-06-04-279>
3. Elliott, M.B., Eastin, B., Caves, C.M.: Graphical description of the action of clifford operators on stabilizer states. *Phys. Rev. A* **77**, 042307 (2008). <https://doi.org/10.1103/PhysRevA.77.042307>
4. Gottesman, D.: Stabilizer codes and quantum error correction. California Institute of Technology (1997)
5. Gottesman, D.: The heisenberg representation of quantum computers. arXiv preprint quant-ph/9807006 (1998)
6. Hein, M., Dür, W., Eisert, J., Raussendorf, R., den Nest, M., Briegel, H.J.: Entanglement in graph states and its applications (2006). arXiv preprint quant-ph/0602096 (2006)
7. Hu, A.T., Khesin, A.B.: Improved graph formalism for quantum circuit simulation. *Phys. Rev. A* **105**, 022432 (2022). <https://doi.org/10.1103/PhysRevA.105.022432>
8. Kapshikar, U., Kundu, S.: On the hardness of the minimum distance problem of quantum codes. *IEEE Transactions on Information Theory* **69**(10), 6293–6302 (2023)
9. Khesin, A.B., Lu, J.Z., Shor, P.W.: Universal graph representation of stabilizer codes. *PRX Quantum* **6**(4) (2025). <https://doi.org/10.1103/1gjs-2rhx>
10. Nethercote, N., Stuckey, P.J., Becket, R., Brand, S., Duck, G.J., Tack, G.: Minizinc: Towards a standard cp modelling language. In: *International conference on principles and practice of constraint programming*. pp. 529–543. Springer (2007)
11. Nielsen, M.A., Chuang, I.L.: *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press (2010)
12. Schlingemann, D.: Stabilizer codes can be realized as graph codes. *Quantum Info. Comput.* **2**(4), 307–323 (2002). <https://doi.org/10.48550/arXiv.quant-ph/0111080>
13. Van De Wetering, J.: ZX-calculus for the working quantum computer scientist. arXiv preprint arXiv:2012.13966 (2020)